GERADOR DE CÓDIGO A PARTIR DE ESPECIFICAÇÕES COM PADRÕES DE REQUISITOS

Aluno(a): Leandro Vilson Battisti

Orientador: Joyce Martins

FURB
UNIVERSIDADE DE BLUMENAU

Agenda I - teoria

- Motivação
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos

Motivação

"A área de requisitos de software é uma grande fonte causadora de problemas no desenvolvimento de software. Talvez por [...] problemas de comunicação com os principais envolvidos no projeto**, descrição ruim dos requisitos** [...]" (GRAHL; DECARLE, 2008, p.1).

Herrington (2003, p. 99) afirma que alguns engenheiros acham que construção de **interfaces de usuário são cansativas e com muitas variações**.

Motivação

- Então...
  - Por que não reduzir a quantidade de interfaces de usuário geradas manualmente gerando diretamente a partir dos requisitos levantados com os *stakeholders*?

Objetivos

O objetivo deste trabalho é gerar código para interfaces gráficas de softwares a partir de especificações escritas utilizando padrões de requisitos.

Objetivos

Os objetivos específicos do trabalho são:

a) processar requisitos baseados em padrões de requisitos de informação;
b) processar requisitos escritos em língua portuguesa com vocabulário irrestrito;
c) processar componentes visuais mais comumente usados na construção de interfaces gráficas;
d) gerar arquivo de definição JSON correspondente à interface gráfica descrita;
e) gerar código fonte a partir do arquivo de definição JSON em pelo menos uma linguagem de programação para validar a ferramenta.

Fundamentação Teórica

- Geração automática de código
- Padrões de requisitos
- Processamento de linguagem natural (PLN)

Geração automática de código

- Geradores passivos
- Etapas
  1. identificar a saída
  2. definir a entrada
  3. interpretar e recuperar as informações
  4. gerar os arquivos de saída

Padrões de requisitos

- 37 tipos divididos em 8 domínios
- Tipos utilizados
  - Tipo de dados (Informação)
  - Estrutura de dados (Informação)
  - Entidade ativa (Entidade de dados)

Tipo de dados

nome	definição
nome	Deve ter no máximo 50 caracteres.
data de nascimento	Deve ser maior que 31/12/1950.
sexo	Será representado por uma letra: “F” ou “M”.

Estrutura de dados

nome	definição
nome pessoal	Os detalhes do nome de uma pessoa são formados pelos seguintes itens de informação: • primeiro nome; • nome do meio; • sobrenome; • iniciais; • título.

Fonte: adaptado de Withall (2007, p. 96).

Entidade ativa

nome	definição
cliente	O sistema deve armazenar as seguintes informações sobre o cliente: • identificação de cliente; • senha; • contato pessoal (definido anteriormente); • data de nascimento; • data de registro; • estado (ativo, bloqueado ou terminado), nunca é apresentado para o cliente. Cada cliente é unicamente definido pelo pela identificação do cliente.

Fonte: adaptado de Withall (2007, p. 130).

Processamento de linguagem natural

- Análise morfológica
  - Reconhecimento de morfologia de palavras conforme contexto

  Exemplo:

  - A **casa** é do João. (é um substantivo)
  - João **casa** amanhã. (é um verbo)
- Dicionário digital x Léxicos computacionais

Trabalhos Correlatos

- Paradigma, por Silva e Martins (2008)
- Ferramenta para geração de classes a partir de PLN, por Baghat et al. (2012)
- Delphi2Java-II, por Silveira (2006)

Agenda II - Desenvolvimento

- Requisitos
- Especificação
- Implementação
- Operacionalidade
- Resultado e discussões
- Conclusões e sugestões

Requisitos

RNF001: Utilização de padrões de requisitos

Nome	**Utilização de padrões de requisitos.**
Tipo do requisito	Aderência a padrão.
Objetivo	Delimitar os padrões de requisitos do protótipo conforme um modelo específico.
Descrição	O protótipo deve ser baseado nos padrões de requisitos tipo de dados, estrutura de dados e entidade ativa, propostos por Withall (2007).

Requisitos

RNF010: Formato do arquivo de saída

Nome	**Formato do arquivo de saída.**
Tipo do requisito	Tecnologia.
Objetivo	Determinar o formato do arquivo de saída gerado pelo protótipo.
Descrição	O protótipo deve gerar como saída um arquivo do tipo JSON ou XML.

Requisitos

RF018: Cadastro do requisito

Nome	**Cadastro do requisito.**
Tipo do requisito	Entidade ativa.
Objetivo	Armazenar as informações sobre requisitos.
Descrição	O protótipo deve armazenar as seguintes informações sobre os requisitos: identificação do requisito e requisito.

Requisitos

RF022: Gerar arquivo de definição

Nome	**Gerar arquivo de definição.**
Tipo do requisito	Transação.
Objetivo	Descrever a geração de arquivos de definição a partir de um formulário.
Descrição	Deve ser possível gerar um arquivo de definição para um formulário. Tanto o formulário a ser utilizado assim como o tipo do arquivo de definição a ser gerado, que pode ser XML ou JSON, deve ser especificado pelo usuário.

Requisitos

RF012: Gerar HTML a partir do arquivo de definição

Nome	**Gerar HTML a partir de arquivo de definição.**
Tipo do requisito	Transação.
Objetivo	Descrever a geração de HTML a partir de um arquivo de definição.
Descrição	Deve ser possível gerar um arquivo HTML a partir do arquivo de definição previamente gerado pelo protótipo.

Especificação

Diagrama de Caso de Uso

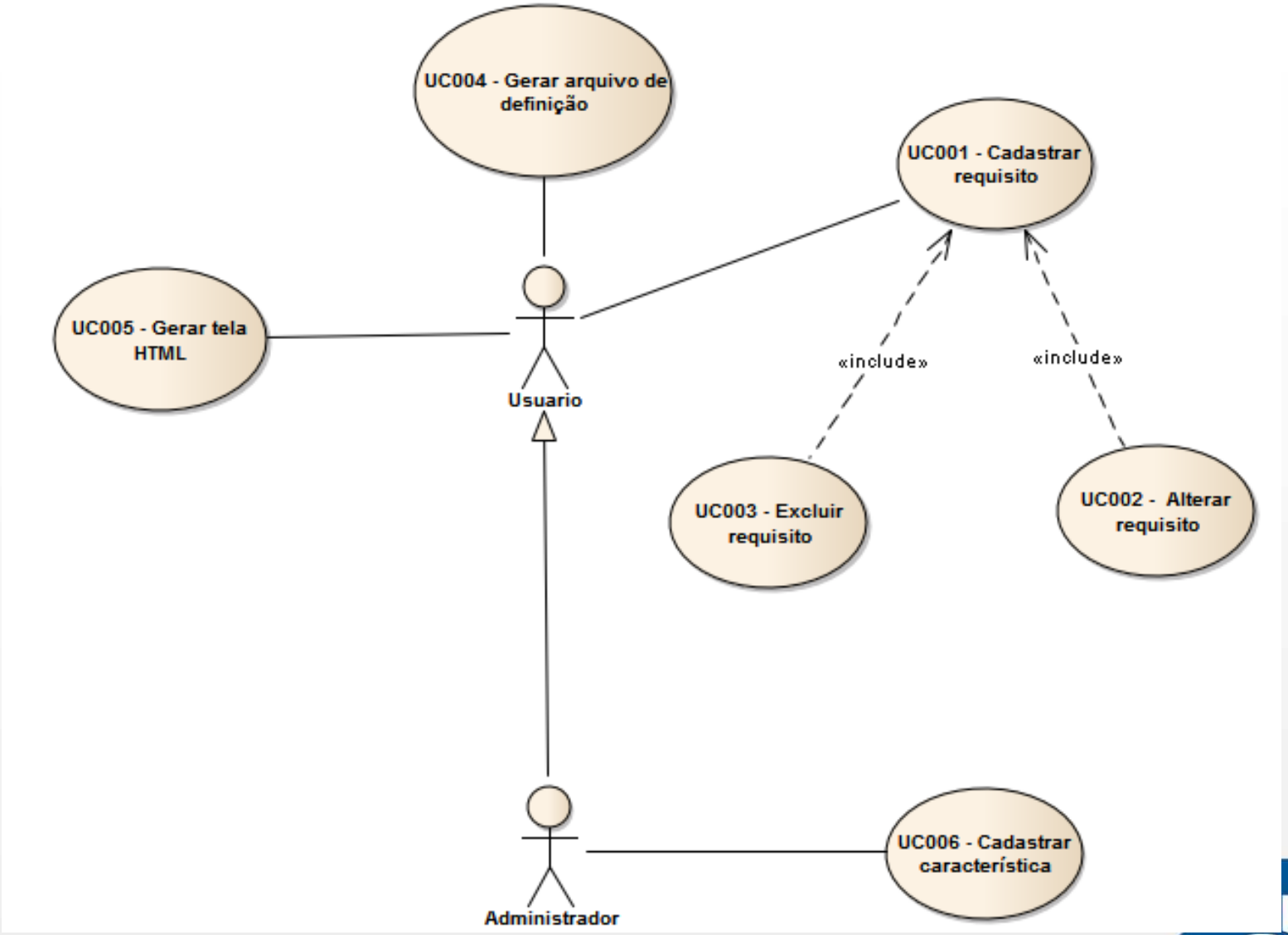


Diagrama de Pacotes

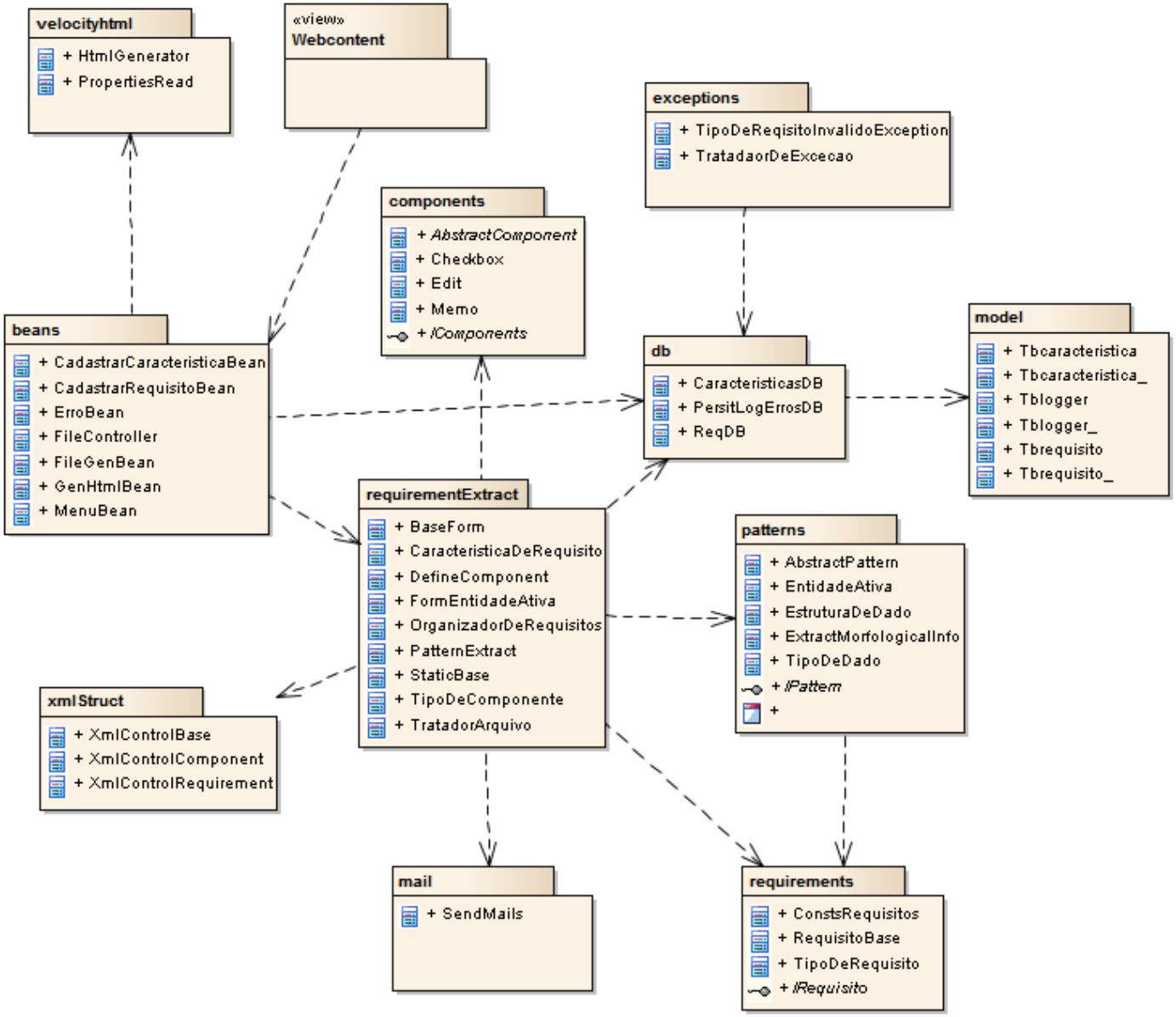
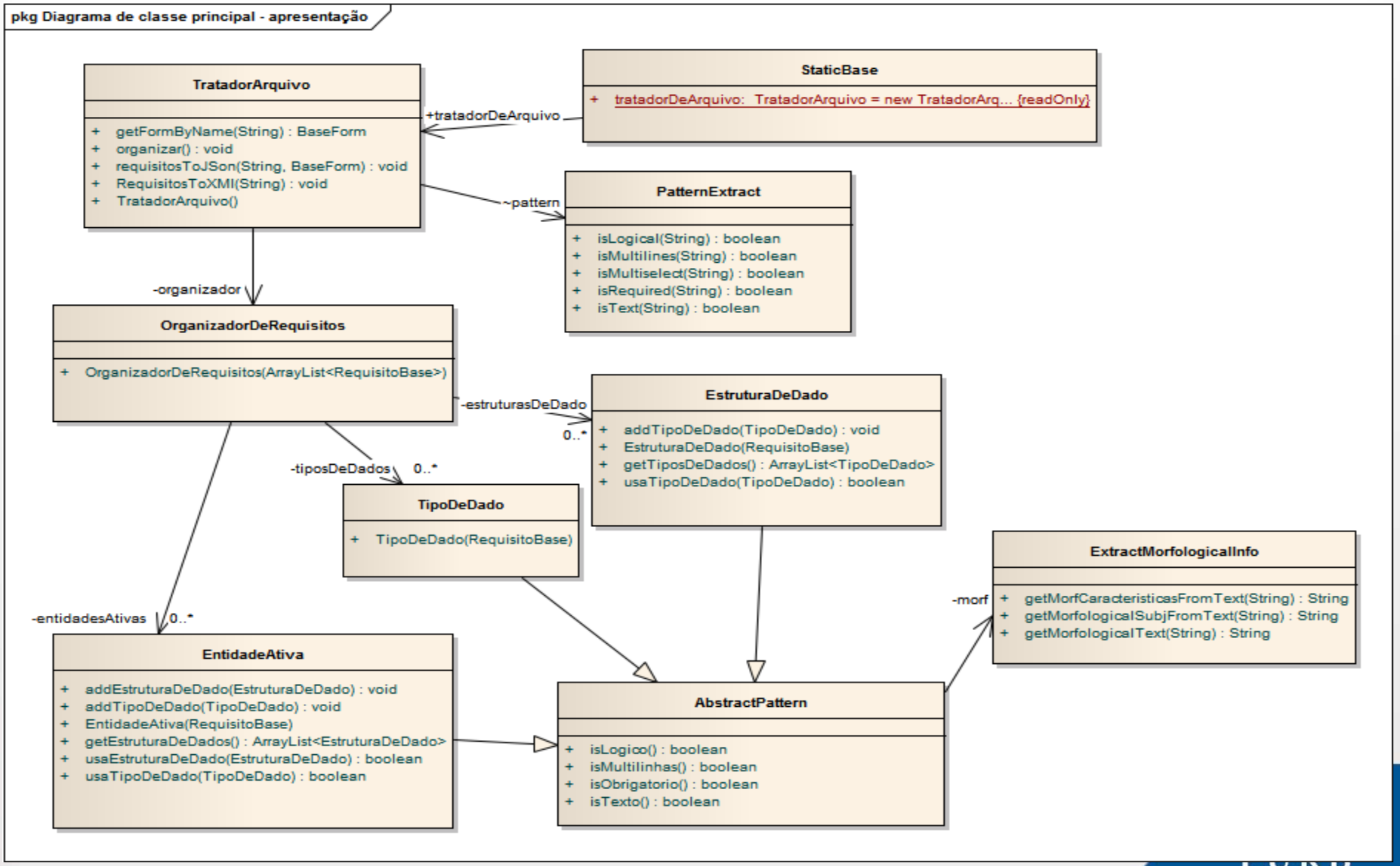
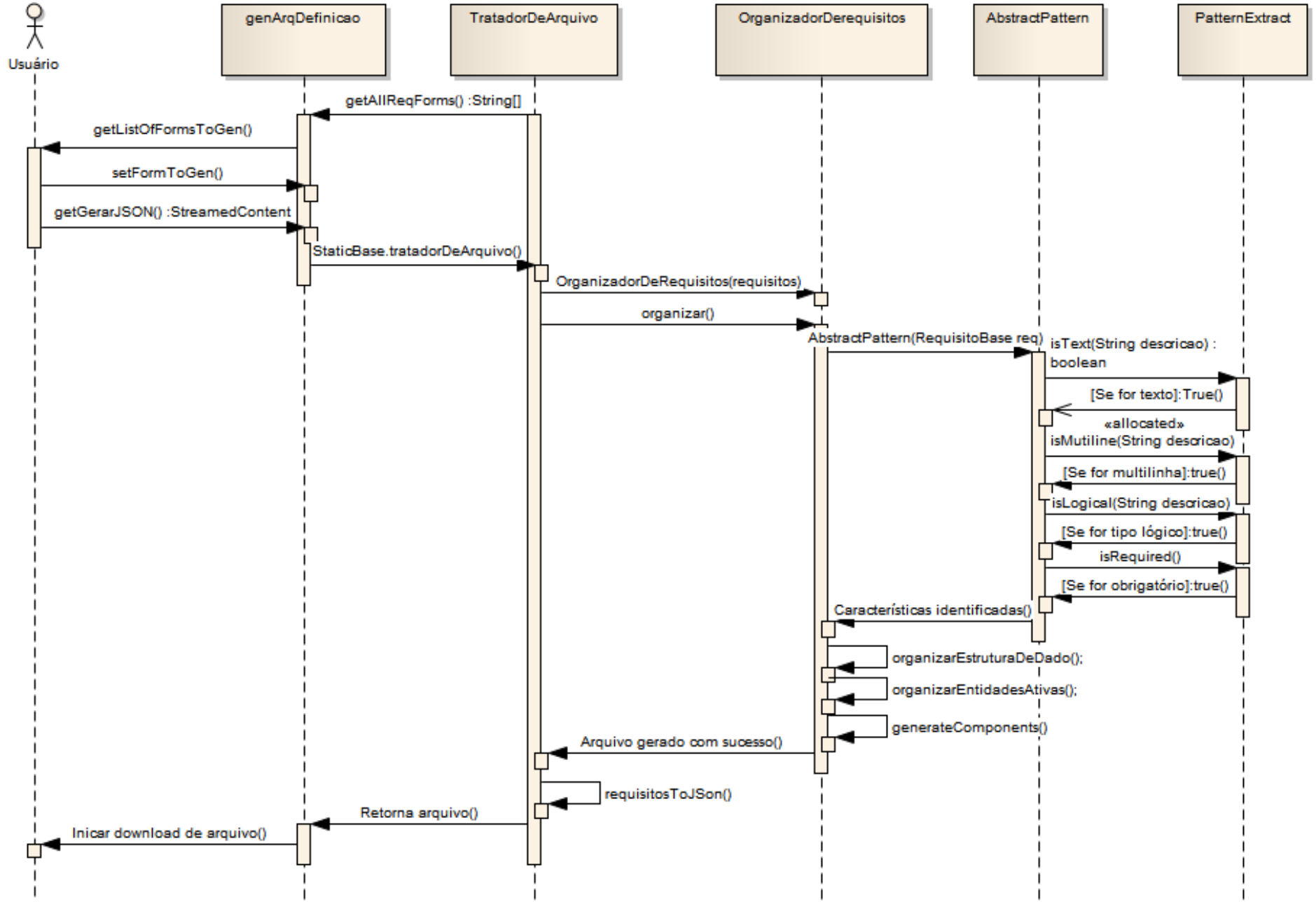


Diagrama de Classes




Usuário
genArqDefinicao
TratadorDeArquivo
OrganizadorDerequisitos
AbstractPattern
PatternExtract
getAllReqForms() :String[]
getListOfFormsToGen()
setFormToGen()
getGerarJSON() :StreamedContent
StaticBase.tratadorDeArquivo()
OrganizadorDeRequisitos(requisitos)
organizar()
AbstractPattern(RequisitoBase req)
isText(String descricao) : boolean
[Se for texto]:True()
«allocated»
isMutiline(String descricao)
[Se for multilinha]:true()
isLogical(String descricao)
[Se for tipo lógico]:true()
isRequired()
[Se for obrigatório]:true()
Características identificadas()
organizarEstruturaDeDado();
organizarEntidadesAtivas();
generateComponents()
Arquivo gerado com sucesso()
requisitosToJSon()
Retorna arquivo()
Inicar download de arquivo()

Implementação

- Técnicas e ferramentas
  - CoGrOO
  - Extração das características de requisitos
  - Definição dos componentes a partir das características
  - Geração do arquivo HTML com Velocity

CoGrOO

- Biblioteca de correção gramatical mantida pela USP (licença GPL)
- Recebe um texto e retorna *tokens* com descrição morfológica

Exemplo:

O rato roeu a roupa do rei de Roma.

– Isto retorna ...

CoGrOO

Sentença: O rato roeu a roupa do rei de Roma			
Tokens:			
o	[o]	art	M=S
rato	[rato]	n	M=S
roeu	[roer]	v-fin	PS=3S=IND
a	[o]	art	F=S
roupa	[roupa]	n	F=S
de	[de]	prp	-
o	[o]	art	M=S
rei	[rei]	n	M=S
de	[de]	prp	-
Roma	[Roma]	prop	F=S

Extração das características de requisitos

- Utilização de análise morfológica (substantivo+adjetivo) e léxicos computacionais

descrição do requisito	característica	exemplo de uso
texto, alfanumérico, numérico, caracteres	texto	O requisito é texto.
lógico	lógico	campo lógico.
texto longo, multilinha, texto de múltiplas linhas	multilinha	É um texto longo.
campo obrigatório	obrigatório	obrigatório.
campo opcional	opcional	opcional.

Definição dos componentes de interface a partir das características

característica	componente
texto	edit
lógico	checkbox
multilinha	memo
obrigatório	esta característica é combinada com as anteriores para informar que o preenchimento é obrigatório
opcional	esta característica é combinada com as anteriores para informar que o preenchimento é opcional

Arquivo JSON

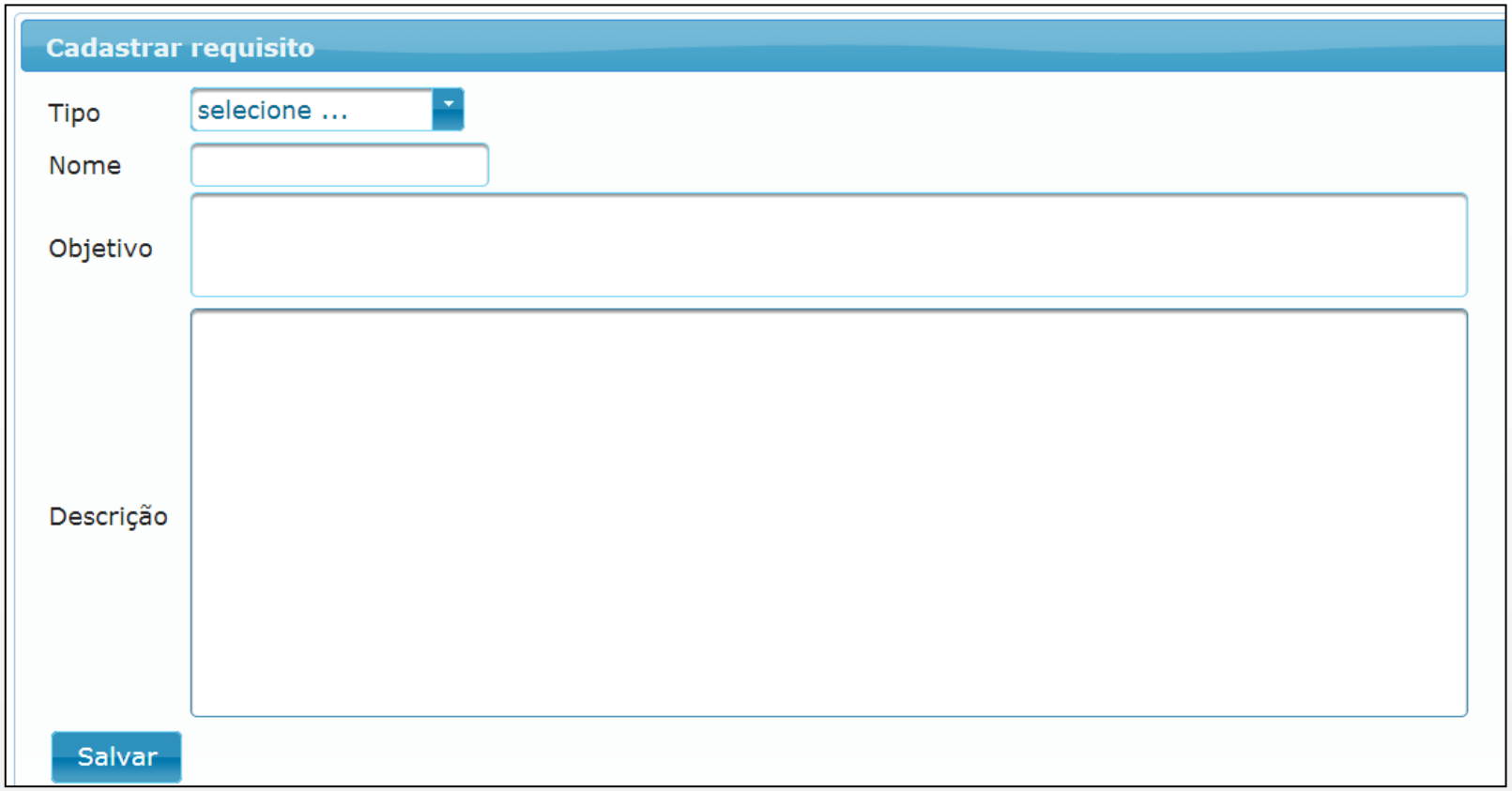
```
{"formularios":
[{"formulario0":{
        "Titulo":"Residência",
        "componentes": {
                "componente1": {
                        "tipo":"tcEdit",
                        "nome":"edrua",
                        "obrigatorio":true,
                        "descricao":"rua"},
                "componente0": {
                        "tipo":"tcCheckbox",
                        "nome":"cbcasapropria",
                        "obrigatorio":false,
                        "descricao":"casa própria"},
                "componente2": {
                        "tipo":"tcEdit",
                        "nome":"edBairro",
                        "obrigatorio":false,
                        "descricao":"Bairro"}
}}}] }
```

Geração do HTML através de Velocity

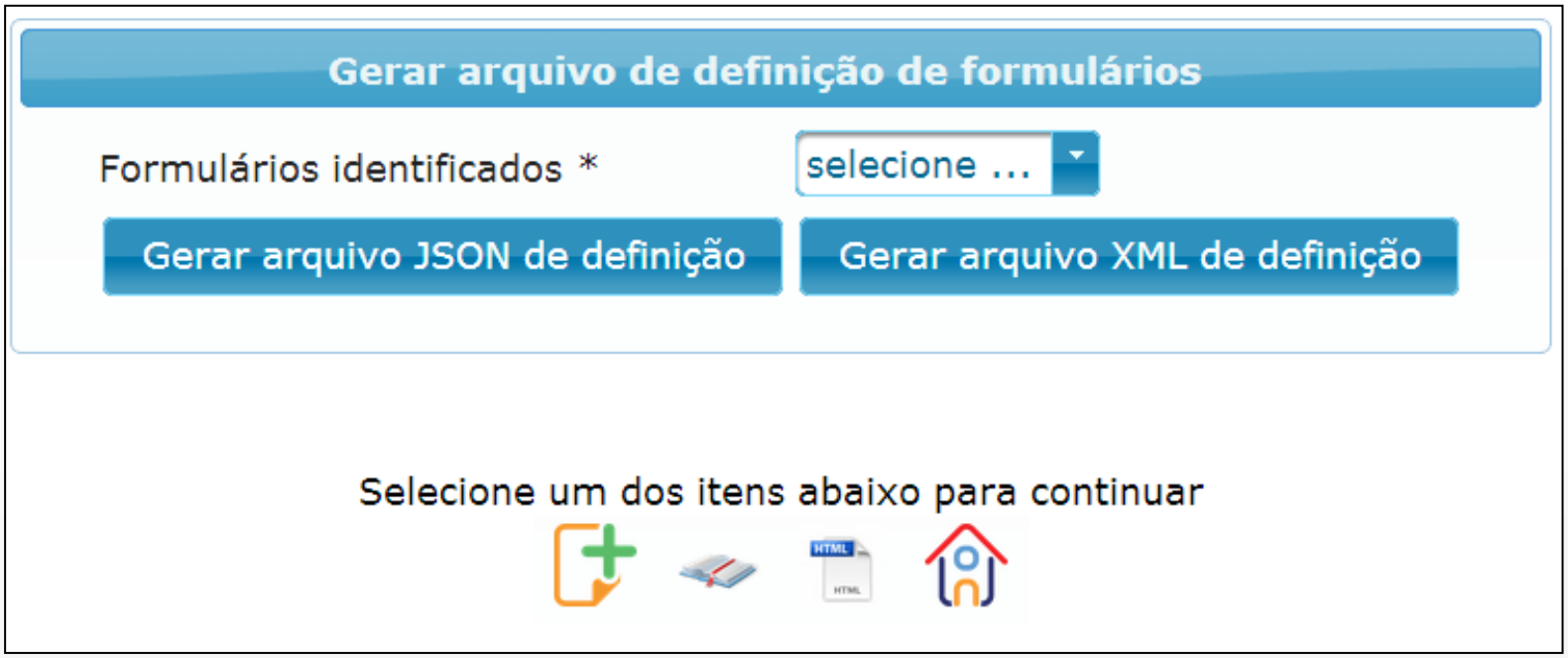
```
#foreach ($comp in $componentes)
  #if($comp.getTipo() == "tcEdit")
    <li>
      <label id="$comp.getNome()">
        $comp.getDescricao()
        #if($comp.getObrigatorio() == "sim")
          <span >*</span>
        #end
      </label>
      <div>
        <span>
          <input type="text"size="10" name="$comp.getNome"/>
         </span>
      </div>
    </li>
   #end
#end
```

Operacionalidade da Implementação

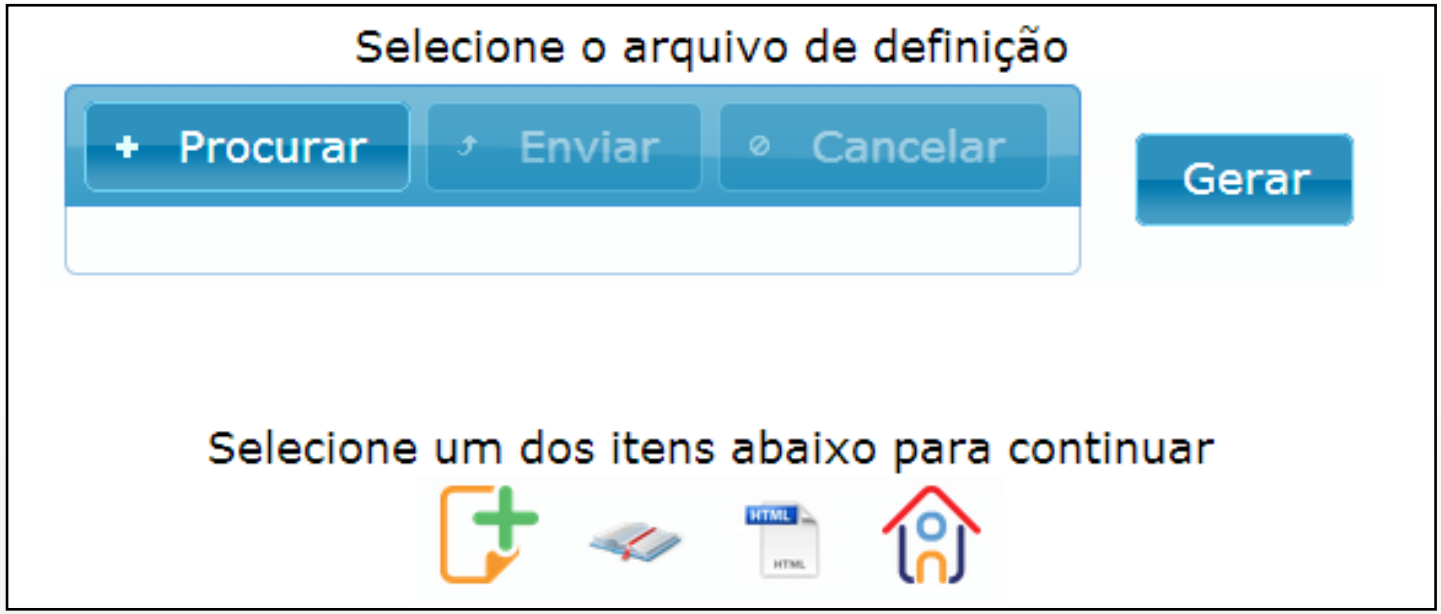
Cadastrar requisito



Gerar definição



Gerar HTML


Selecione o arquivo de definição
Procurar
Enviar
Cancelar
Gerar
Selecione um dos itens abaixo para continuar
HTML
HTML

Resultados e Discussões

- Disponibilizado para usuários finais sob a URL http://elicitar.elasticbeanstalk.com/ em 26 de maio de 2014
- Reconhecimento limitado ao dicionário de léxicos computacionais
- Gera arquivo de marcação (HTML) e não código de alguma linguagem

Principais resultados da pesquisa

Nº	Pergunta	Respostas
2	Em sua percepção qual o percentual de telas **CRUD** (Create, Read, Update, Delete), ou seja, telas simples de cadastro, de suas aplicações?	entre 41% e 60%: 44% das respostas
5	O arquivo HTML gerado pela ferramenta Elicitar foi conforme o esperado?	**próximo do desejado**: 44% das respostas
8	Quanto você acredita que uma ferramenta ou *framework* para gerar as telas da aplicação diretamente a partir da especificação **aumenta a produtividade**?	entre 41% e 60%: 33% das respostas acima de 80%: 33% das respostas
9	De 0 a 5, qual a sua nota geral para a ferramenta Elicitar?	nota 4: 67% das respostas

Comparativo com correlato

característica	ferramenta Elicitar	Bhagat et al. (2012)
plataforma	web	desktop
linguagem natural reconhecida	português*	inglês*
formato do texto de entrada	requisitos escritos em padrões de requisitos	texto livre
formato de saída	XML ou JSON com definições	texto livre

Conclusões e Sugestões

- Obteve bom resultado de detecção dentro do que a ferramenta se propõe. O reconhecimento obteve um bom resultado segundo usuários.
- Necessária a transformação de requisitos “normais” em padrões de requisitos.
- Reconhecimento restrito aos léxicos computacionais cadastrados.
- Disponibilizar a ferramenta para usuários testarem livremente e opinarem ajudou bastante em correções de erros e avaliação da ferramenta.

Extensões

- Suporte a outros tipos de padrões de requisitos (domínio de funções de usuário).
- Suporte a novos componentes.
- Implementar uso de SBVR como dicionário léxico computacional para identificação de características.
- Gerar código fonte em linguagem de programação a partir do arquivo de definição.
- Transformar o protótipo em *plugin* para IDEs como Eclipse e Genexus.

Demonstração